CHAPTER 4

THE APPLICATIONS OF METAHEURISTICS IN INTEGRATED INVENTORY ROUTING PROBLEM

4.1 Introduction

Inventory Routing Problem (IRP) is the coordination of the inventory management and transportation, where both are vital components of the Supply Chain Management (SCM). The importance of integration and coordination of different components in SCM has been widely acknowledged as it can be an advantage for many companies in a competitive business approach. The IRP seeks to determine simultaneously an optimal inventory and distribution strategy that minimizes the total cost. The resulting inventory and transportation policies usually assign suppliers to routes and then determine the replenishment intervals and collection sizes for each supplier. The implementation of IRP is critical especially in a Vendor Managed Inventory (VMI) replenishment system where the supplier or manufacturer observes and controls the inventory levels of its customers or retailers. In VMI, the supplier is fully responsible in managing the customer's inventory level. This is a symbiotic relationship. VMI makes it less likely that a business will unintentionally become out of stock of a good and reduces inventory in the supply chain. Furthermore this is most beneficial if the holding cost at the vendor is far greater than the holding cost at the retailers. The availability of data and information systems through the advances in the technology and communications system has made it possible for the implementation of VMI. One of the most important benefits of the VMI is that it permits a more uniform utilization of transportation resources. This leads to a higher level of efficiency and a much lower distribution cost which often constitutes the largest part of the overall cost. This operation is best illustrated in the following Figure 4.1 (Waters, 2003).

There are several variants of IRP, depending on the underlying assumptions made in the model. Many models look at an infinite horizon where the transportation cost is approximated as the average cost, reflecting the long term nature of the problem. Recently, many researchers have started to consider finite horizon in which the solutions are valid within certain planning horizons only. This is often true when the demand cannot be approximated by certain functions or when the demand varies considerably from one period to another. Finite horizons can be further classified as a single period or multi periods. Other factors that may influence the model may include the number of items considered, the type of product (whether it is a single product or multi product), and the demand of the product (whether it is constant, time varying, deterministic or stochastic in nature). In this research, the focus was on a finite horizon model, of multi periods with multi products and the demand for each product was deterministic, time varying and product dependent.

Another important factor to take into consideration is the type of distribution network to be considered. Much of the traditional logistics focuses on the outbound distribution of products to centres, retailers or customers. However, inbound logistics plays an important role in some industries, especially in automotive industries where many manufacturers have chosen to modularize the vehicle design and outsource the production of more component parts and modules. This has created an environment where managing the inbound logistics of these modules and parts from a large number of different suppliers effectively has become most vital. Inbound logistics is not limited to automotive industries but it is also an important problem for any business that has a large number of low to medium volume suppliers where shipment consolidation offers an opportunity for total cost savings (Stacey *et al.*, 2007). An illustration of the movement of goods in both logistics is given in Figure 4.2.

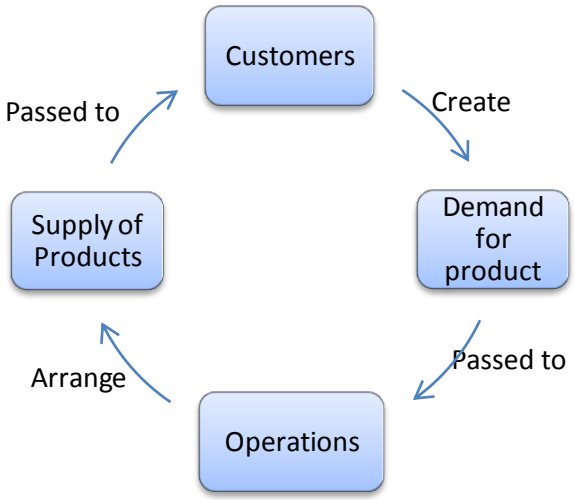


Figure 4.1: Cycle of supply and demand.

The focus now will be on inbound logistics where distinct spare parts are collected from the suppliers.

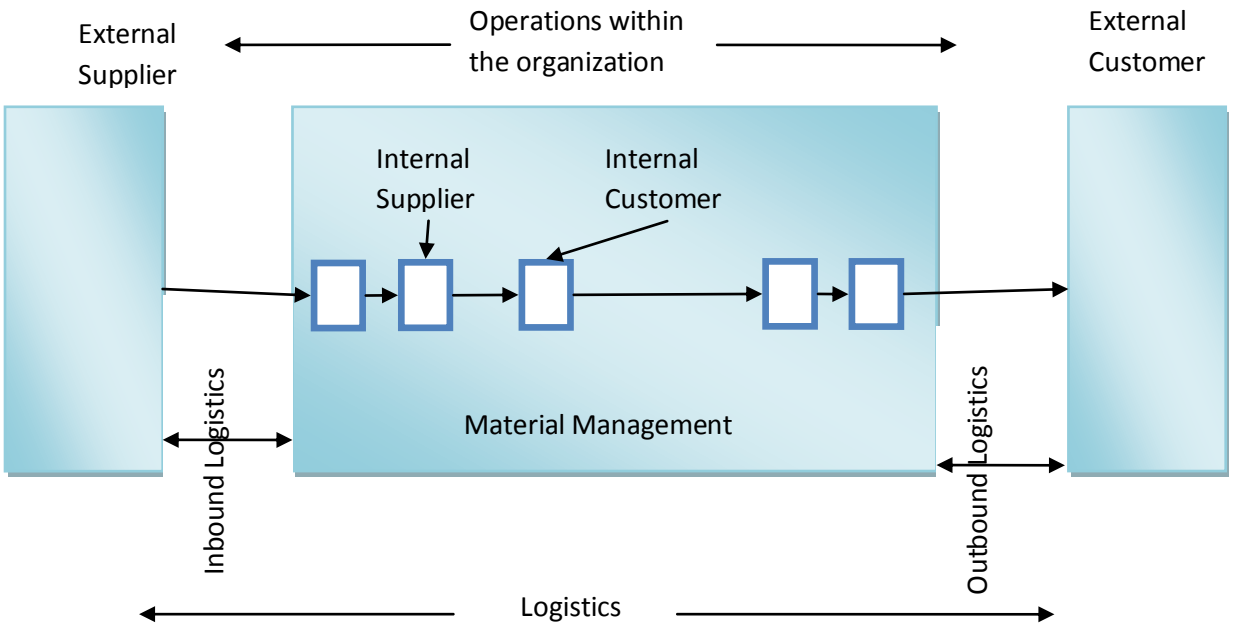


Figure 4.2: The role of logistics.

4.2 Literature Review

The coordination of inventory and routing was first considered in Federgruen and Zipkin (1984) where the problem was treated as a single day problem with a limited amount of inventory and the customers' demands were assumed to be a random variable. The problem is modeled as a nonlinear integer program using a generalized Benders' decomposition approach. This approach has the attributes that for any assignment of customers to routes, the problem decomposes into a nonlinear inventory allocation problem which determines the inventory and shortage costs and a TSP for each vehicle considered which produces the transportation cost. This is an extension of the classical vehicle routing problems, in which this model integrates vehicle routing and inventory control problems. However, not all customers will be visited every day as the inventory and shortage costs as well as the limited amount of inventory are to be considered. Federgruen *et al.* (1986) expanded the idea for the perishable product problem. The product in the system was divided into two classes. Old units was a class where the product would perish in the present period while fresh units were those with at least one period away from their perish date. The solution was adopted from their earlier work (Federgruen and Zipkin, 1984) with a variation that the inventory allocation sub-problems accounted for two product classes.

Chien *et al.* (1989) were among the first to simulate a multiple period planning model based on a single period approach. This was achieved by passing some information from one period to the next through an inter-period inventory flow. In their problem, there was a central depot with all customers assigned to it. The supply capacities of the depot and the demand of the customers were fixed. An integer program was modeled using a Lagrangean dual ascent method to handle the allocation of the limited inventory available at the plant to the customer, the customer to vehicle assignment, and the routing of the vehicle.

Fisher *et al.* (1982) studied a real-time problem of inventory routing at Air Products which is an industrial gas producer. The objective was to maximize the profit from product distribution over several days. The demand was given by the upper and lower bounds on the amount to be delivered to each customer for every period in the planning horizon. A similar approach as the one proposed by Chien *et al.* was adopted.

Dror and Ball (1987) and Dror *et al.* (1986) both considered the effects of the short-term planning period in the subsequent decisions. They proposed an integer program where the consequences of the present decisions on later periods were accounted for by using penalty and incentive factors. In this problem, the single period models were used as sub-problems. Besides using a similar analysis, Dror and Levy (1986) yielded a weekly schedule by introducing a post optimization based on node and arc exchanges.

Dror and Trudeau (1990) successfully demonstrated split delivery. Split delivery is where the demand of a customer can be supplied by more than one vehicle; this can reduce the travel distance and also the number of vehicles needed. On the other hand, the complexity of the problem also increased (Dror and Trudeau, 1990).

Bertazzi *et al.* (1997) proposed a set of decomposition heuristics for the transportation of multi product in a multi period with a constant demand. Starting from a link by link solution of the problem (i.e. direct shipping), a local search was performed looking for improvement through consolidation. The second phase aggregated customers who visited at the same frequency on the same route. The authors introduced the concept of split deliveries where the quantity of a product required at a destination may be split between different shipments, possibly with different frequencies. For simplicity, most multi-product models assumed that each retailer required only one type of product. It is noted that the same assumption is adopted in this

model where each supplier is assumed to supply a distinct product to the assembly plant.

Bertazzi *et al.* (2002) considered a multi-period model with deterministic demand in which a set of products was shipped from a supplier to several retailers. They adopted the order up to a level management inventory, (S, s), where the maximum (S) and the minimum (s) levels of inventory were specified by each retailer and the products needed to be replenished before the minimum level was attained. The quantity of the product delivered was the amount such that the maximum level was reached at the retailer. The authors proposed a two stage heuristic algorithm whereby the first stage focused on route construction algorithms whilst the second stage attempted to improve the existing solution by performing simple swap operators that aimed in removing or inserting customers at different positions of the route used by the vehicle. This solution procedure was implemented for only a single product and a single vehicle case though the authors suggested that their approach could easily be modified to cater for the multi product case.

Lee *et al.* (2003) tackled a class of IRP where there were multiple suppliers and an assembly plant in an automotive part supply chain. They addressed the problem as a finite horizon, multi-period, multi-supplier, and single assembly plant part-supply network. The objective of their study was to minimize the total transportation and inventory cost over the planning horizon. The problem was divided into two sub-problems that was vehicle routing and inventory control. To solve these problems, a mixed integer programming model was proposed using a heuristics based on simulated annealing. The purpose of using the heuristics was to generate and evaluate alternative sets of vehicle routes while a linear program determined the optimum inventory levels for a given set of routes. The authors also claimed that the optimal solution was dominated by the transportation cost, regardless of the magnitude of the unit inventory

carrying cost. Here, it was assumed that no backordering was allowed since any shortage of parts led to excessively high costs at the assembly plant.

Ribeiro and Lourenço (2002) investigated an IRP model for two types of customers namely the VMI customers and the customer managed inventory (CMI) customers. The former customers had a random demand and the distributor managed the stock at the customers' location. Meanwhile, the CMI type of customers had a fixed demand and there were no inventory costs for the distributor. The results showed that the inventory and transportation management in an integration model yielded a better performance than its counterpart in the non integrated case.

Yu *et al.* (2008) solved a large scale inventory routing problem with split delivery (IRPSD) by proposing an approximate model and Lagrangian relaxation. Linear programming and a minimum cost flow were used to solve the relaxed problem in which the problem was decomposed into an inventory problem and a routing problem. It is noted that the demand was deterministic and there was no back ordering allowed.

Huang and Lin (2010) proposed a modified ACO for a multi item inventory routing problem where the demand was uncertain. The algorithm was to determine an optimal inventory policy and replenishment route. Varying from the traditional ACO, their algorithm also calculated neighbours that made use of the pheromone attraction on nodes. Their model of IRP was mainly based on the vending machine replenishment environment.

Several IRP with stochastic demands were also studied; for example Campbell *et al.* (1998) and Trudeau and Dror (1992) and together with the literature reviews given in Campbell *et al.* (2002), Campbell and Savelsbergh (2004) and Kleywegt *et al.* (2002a, 2002b, 2004). For a comprehensive review of the IRP the readers are referred to Moin and Salhi (2007).

4.3 Problem Description

A distribution network consisting of a depot, an assembly plant and N geographically dispersed suppliers were considered. Each supplier supplied a distinct product to the assembly plant to meet the demand in each period. The problem addressed here was based on a finite horizon, multi-period, multi-supplier, and single assembly plant, where a fleet of capacitated vehicles housed at a depot transported products from the suppliers to meet the demand specified by the assembly plant. At the end of each trip, vehicles returned to the depot. No backordering/backlogging was allowed. However, if the demand for more than the period's demand was collected, the inventory was carried forward subjected to the product-specific holding cost incurred at the assembly plant, where the demand was assumed to be ready when the vehicle arrived. Note that the pick-up quantity should not exceed a vehicle's capacity as the product was not allowed to be split between different vehicles. The objective was thus to minimize the overall transportation and inventory carrying costs over the planning horizon. Both the transportation and inventory carrying costs were considered concurrently.

Figure 4.3 illustrates the case of 5 suppliers and 2 periods. In this example, the vehicle capacity was considered to be 10. Note that though supplier number 3 required 2 units in period 1 and 5 units in period 2, a feasible and efficient schedule (depending on other cost) was to collect 4 units in period 1and then collect the remainder 3 units in period 2, thus saving on the capacity cost for this supplier.

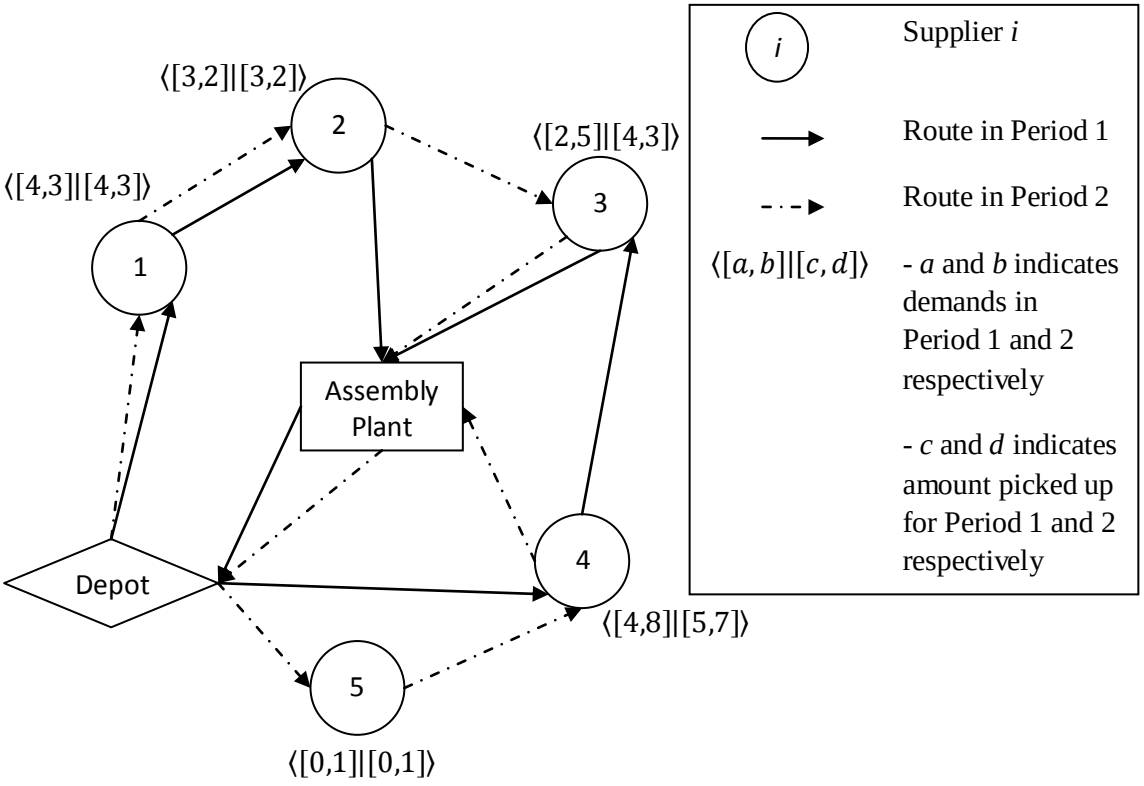


Figure 4.3: An example of pick-up routes for the 5 retailer and 2-period problems.

4.4 Problem Formulation

Several assumptions have been made in the model. The assumptions were as follows:

1. No backordering was allowed as the cost would be too excessive.
2. Split delivery between different vehicles was not allowed. Any supplier could only be visited by at most one vehicle.
3. Quantity collected by each vehicle could not exceed a vehicle's capacity.
4. Product specific holding costs at the assembly plant were given for each part type and inventory cost at the suppliers was not considered. The product was assumed to be ready when the vehicle arrives.
5. The number of capacitated vehicles available at the depot was unlimited and each vehicle had to return to the depot after the completion of each route.
6. The positions of the depot, assembly plant, and suppliers were given and fixed.

7. The route length for any vehicle was fixed and could not exceed a user specified limit.
8. Transportation costs included a fixed cost for each trip or every time a vehicle was initiated plus a variable cost proportional to the travel distance.
9. Planning horizon was finite and given.

A mathematical formulation based on Lee *et al.* (2003) is presented. The following notations are introduced first as follows:

Indices

$S = \{1,2,\ldots,N\}$	A set of suppliers where supplier i $(i \in S)$ supplies product i only.
$D = \{0\}$	Depot
$P = \{N+1\}$	Assembly plant
$\tau = \{1,2,\ldots,T\}$	Period index

Parameters

C	Vehicle capacity
F	Fixed vehicle cost per trip (assumed to be the same for all periods)
V	Travel cost per unit distance
M	Size of the vehicle fleet and it is assumed to be ∞ (unlimited)
L	Route Length
J_t	An upper bound on the number of vehicles (trip) in period t
d_{it}	Demand for product from supplier i (at the Assembly Plant) in period t
c_{ij}	Travel distance between supplier i and j where $c_{ij} = c_{ji}$ and the triangle inequality, $c_{ik} + c_{kj} \geq c_{ij}$, holds for any i, j, k with $i \neq j, k \neq i$ and $k \neq j$

h_i	Inventory carrying cost at the Assembly Plant for product from supplier i per unit product per unit time
I_{i0}	Initial inventory level of product from supplier i (at the Assembly Plant) at the beginning of period 1

Variables

a_{it}	Total amount to be picked-up at supplier i in period t
I_{it}	Inventory level of product from supplier i at the assembly plant at the end of period t

The mathematical formulation for the IRP can be expressed as:

$$Z = \min \quad V \sum_{i=0}^{m+1} \sum_{j=0}^{m+1} c_{ij} \left(\sum_{t=1}^{T} \sum_{k=1}^{J_t} x_{ijkt} \right) + \sum_{i=1}^{m} h_i \left(\sum_{t=0}^{T} I_{it} \right) + F \sum_{t=0}^{T} \sum_{k=1}^{J_t} y_{0kt} \tag{1a}$$

Subject to:

$$0 \le a_{ikt} \le C \cdot y_{ikt}, \ \forall i \in \{1, \ldots, m\}, \ \forall k, \forall t \tag{1b}$$

$$\sum_i^k a_{ikt} \le C, \ \forall k, \forall t \tag{1c}$$

$$\sum_j \ x_{ijkt} = \sum_j x_{j,i,k,t} = y_{i,k,t} \forall i \in \{1, \ldots, m\}, \quad \forall k, \ \forall t \tag{1d}$$

$$u_{ikt} - u_{jkt} + m \cdot x_{ijkt} \le m - 1, \ \forall i, j \in \{1, \ldots, m\}, \ \forall k, \ \forall t \tag{1e}$$

$$I_{it+1} = I_{it} + a_{it} - d_{it}, \ \forall i \in \{1, \ldots, m\}, \ \forall t \tag{1f}$$

$$x_{i0kt} = 0, \forall i \in \{1, \ldots, m\}, \quad \forall k, \ \forall t \tag{1g}$$

$$x_{m+1jkt} = 0, \forall j \in \{1, \ldots, m\}, \quad \forall k, \ \forall t \tag{1h}$$

$$x_{m+10kt} \ge x_{ijkl}, \forall i, j \in \{1, \ldots, m+1\}, \quad \forall k, \ \forall t \tag{1i}$$

$$\sum_{ij} c_{ij} \cdot x_{ijkt} \le L, \forall k, \ \forall t \tag{1j}$$

$$I_{it} \ge 0, \forall i \in \{1, \ldots, m\}, \ \forall t \tag{1k}$$

$$y_{ikt}, \ x_{ijkt} \in \{0,1\}, \ \forall i, \forall j \in \{1, \ldots, m\}, \forall k, \forall t \tag{1l}$$

The objective function (1a) consisted of fixed plus variable travel cost and also the inventory carrying cost. Constraint (1b) ensures that the amount picked up at each

supplier does not exceed the vehicle capacity. Constraint (1c) ensured that the total picked up does not violate the truck capacity. Constraint (1e) served as the sub-tour elimination constraint for each truck in each period. The inventory balance equation was given by constraint (1f). Constraint (1k) is to make sure that the demand at the assembly plant is completely fulfilled without backorder. Constraint (1j) limit the length of each route to L. Constraint (1l) is the binary decision variables.

Note that a straight forward upper bound on the number of vehicles required in each period can be approximated by $J_t = \left\lceil \frac{\sum_{t=1}^{T} \sum_{i=1}^{N} d_{it}}{C} \right\rceil$. The actual number of trips or vehicles required was often more than J_t as the product could not be served by more than one vehicle. The minimum number of vehicles required was equivalent to the optimal solution of the Bin Packing Problem (Campos *et al.*, 2008). For simplicity of terminology, trips and vehicles were interchangeable. One truck could only perform one trip in each period. Note that the truck could be reused after it returned to the depot. It is also noted that the decision variable x_{ijkt} equaled to 1 if truck k visited supplier i, followed by supplier j in period $t$and equaled to *0* if otherwise. And also y_{ikt} equaled to 1 if truck k visited supplier i in period $t$and equaled to *0* if otherwise.

4.5 Implementation of Metaheuristics for IRP

A Variable Neighborhood Search (VNS) metaheuristic was proposed for solving the IRP. The results were then compared to the Genetic Algorithms that was developed by Aziz and Moin (2007).

4.5.1 Variable Neighbourhood Search (VNS)

VNS is based on the exploration of a dynamic neighbourhood model. As discussed in Chapter 2, VNS works on the principles that different neighbourhoods generate different search topologies (Hansen and Mladenovic, 2001; Blum and Roli,

2003). Systematic change of the neighbourhood is applied within a local search algorithm. This local search may be applied repeatedly in order to move from the incumbent solution.

As mentioned earlier, IRP comprises of two components: the inventory costs which may include the set up cost, and the transportation cost. The transportation cost may consist of the variable and the fixed travelling cost. Therefore, IRP attempts to find a tradeoff between the inventory and the travelling costs. The choice of a suitable neighbourhood structure in IRP is not straight forward. The neighbourhood can be defined as the symmetric difference between the different clusters within the same period or the symmetric difference between the number of customers who visited in each period.

Neighborhood Structure

In this research, the neighbourhood structure k , N_k , $k=1,2,...,k_{\max}$ is defined as a distance function, that is the cardinality of the symmetric difference between any two solutions V_1 and V_2 written as $\rho(V_1,V_2)=|V_1 \setminus V_2|$. The symmetric difference of two solutions is defined as the number of different suppliers that are visited in a period. Neighbourhood structure is determined by comparing the number of different suppliers that are visited in each period in the planning horizon. In the implementation this is achieved, for N_k,by selecting randomly k number of suppliers to be removed. Assuming that the problem is expressed as a binary matrix (as in the GA) where one(1) indicates that the particular supplier will be visited in that particular period and zero(0) otherwise. Without loss of generality, it is assumed that the initial solution is represented by all ones, where collections are made according to the demand in each period. Examples of a neighbourhood structure are illustrated in Figures 4.4a and 4.4b.

Note that the difference between the examples and the initial solution is one in Figure 4.4a.

		Period				
		1	2	3	4	5
Supplier	1	1	1	1	1	1
	2	1	1	1	1	1
	3	1	0	1	1	1
	4	1	1	1	1	1
	5	1	1	1	1	1

		Period				
		1	2	3	4	5
Supplier	1	1	1	0	1	1
	2	1	1	1	1	1
	3	1	1	1	1	1
	4	1	1	1	1	1
	5	1	1	1	1	1

Figure 4.4a: Examples of members in a neighbourhood structure N_1.

		Period				
		1	2	3	4	5
Supplier	1	1	1	1	1	1
	2	1	1	1	0	1
	3	1	0	1	1	1
	4	1	1	1	1	1
	5	1	1	1	1	1

		Period				
		1	2	3	4	5
Supplier	1	1	1	0	1	1
	2	1	1	1	1	0
	3	1	1	1	1	1
	4	1	1	1	1	1
	5	1	1	1	1	1

Figure 4.4b: Examples of members in a neighbourhood structure N_2.

Prior to implementing the VNS, the Double Sweep Algorithm (DSA) is applied for the clustering and routing of suppliers. The DSA sweeps horizontally emphasizing the *x*-coordinates for the clustering of the suppliers, while the *y*-coordinates are used to carry out the routing. A limited experiment carried out shows that the DSA often produces superior results compared to the sweep algorithm of Clarke and Wright (1964). The intensification of the search within each period is achieved through the implementation of the Generalized Insertion (GENI) method proposed by Gendreau *et al.* (1992). GENI is embedded in the local search.

GENI (Generalized Insertion Method)

The GENI procedure has been introduced by Gendreau *et al.* (1992) for the TSP. The TSP usually requires an insertion of a vertex which can be classified as tour construction procedures. Implemented as the tour improvement procedure, the algorithm has to be modified when changing positions of the vertices to produce a feasible tour.

Vertex *v* is introduced first. Note that the insertion of the vertex *v* into the tour not needed to take place between two adjacent vertices is the distinct characteristic of GENI. As proposed by Gendreau *et al.* (1992) there are two types of GENI insertions: Type I Insertion and Type II Insertion. Some explanation on Type I and Type II Insertions are given and illustrated. For a more detailed explanation on the GENI Type I and Type II Insertions can be found in Gendreau *et al.* (1992).

Type I Insertion

There are several assumptions made in Type I Insertion. Those are $v_k \neq v_i$ and $v_k \neq v_j$. Assume that the insertion vertex is v. Inserting vertex$v$ in the tour needs the deletion of arcs (v_i, v_{i+1}), (v_j, v_{j+1}) and (v_k, v_{k+1}). They are then replaced by (v_i, v), (v, v_j), (v_{i+1}, v_k) and (v_{j+1}, v_{k+1}). And also this implies that the two paths $(v_{i+1}, \ldots, v_j)$ and $(v_{j+1}, \ldots, v_k)$ are reversed. The insertion can be pictured as follows: (Figure 4.5a).

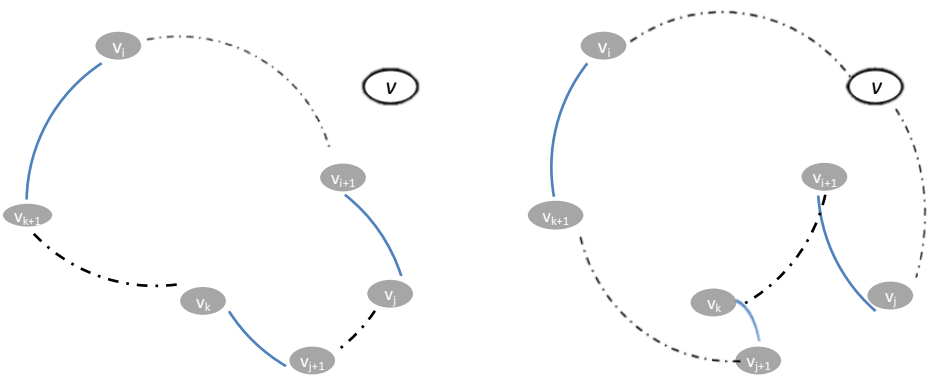


Figure 4.5a: Type I insertion of vertex *v* between v_i and v_j.

Type II Insertion

Similar to Type I, several assumptions are made in Type II Insertion. Those are $v_k \neq v_j, v_k \neq v_{j+1}$, $v_l \neq v_i$and $v_l \neq v_{i+1}$. Assume that the insertion vertex is v. Inserting vertex$v$ in the tour needs the deletion of arcs(v_i, v_{i+1}), (v_l, v_{l-1}), (v_j, v_{j+1}) and (v_{k-1}, v_k). They are then replaced by(v_i, v), (v, v_j), (v_l, v_{j+1}), (v_{k-1}, v_{l-1}) and (v_{i+1}, v_k). And also this implies that the two paths $(v_{i+1}, \ldots, v_{l-1})$ and $(v_l, \ldots, v_j)$ are reversed. The insertion can be pictured as follows: (Figure 4.5b).

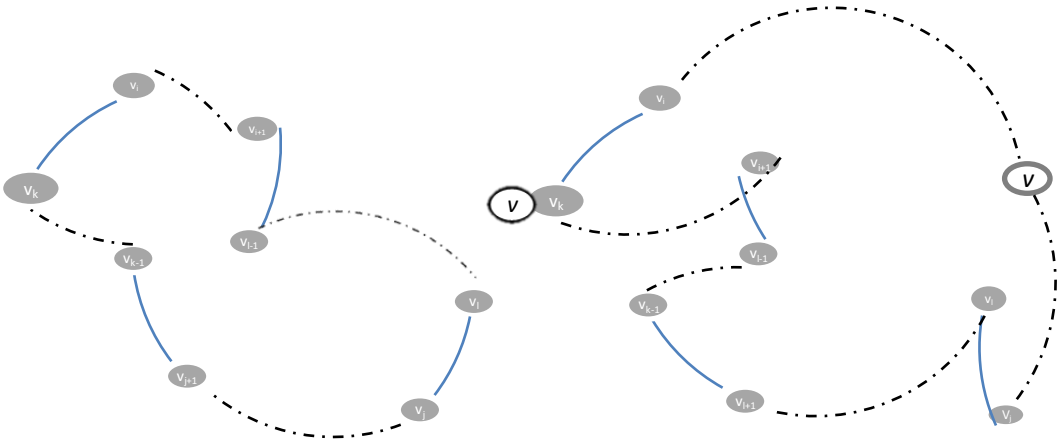


Figure 4.5b: Type II insertion of vertex *v* between v_i and v_j.

GENI Algorithm

Step 1 *Create an initial tour by selecting an arbitrary subset of three vertices. Initialize the p-neighbourhoods of all vertices.*

Step 2 *Arbitrarily select a vertex v not yet on the tour. Implement the least cost insertion of v considering the two possible orientations of the tour and the two insertion types. Update the p-neighbourhoods of all vertices to account for the fact that v is now on the tour.*

Step 3 *If all vertices are now part of the tour, stop. Otherwise go back to Step 2.*

GENI algorithms consider every possible insertion in the two possible orientations of the tour, which are clockwise and counterclockwise. In Gendreau *et al.* (1992), it was argued that because of the potential number of choices for v_i, v_j, v_k, and v_l (only introduced in Type II insertion) on the order of n^4, the search was limited as described in the following section.

For any vertex $v \in V$, define its p-neighbourhood$N_p(v)$ as the set of the p vertices on the tour closest to v (with respect to the c_{ij}'s). c_{ij} can be terms of cost or distance. And if v has fewer than $p$neighbours on the tour, all of them belong to $N_p(v)$. Given parameter p, initially vertexv_i, v_j followsv_k and thenv_l, with $v_i, v_j \in N_p(v)$,$v_k \in N_p(v_{i+1})$, and $v_l \in N_p(v_{j+1})$. With $v_i \in N_p(v)$, all insertions of v will be between two consecutive vertices v_i and v_{i+1} . In practice, p is a relatively small number. c_{ij} is the cost for travelling from city *i* to city *j*.

However, in this research, only GENI Type I insertion was applied. GENI Type II insertion was not suitable as from observation usually after the clustering process of using DSA, the number of customers in a vehicle was not more than 6 and Type IIGENI required at least 8 customers in general. However, sometimes (but very rare) it was observed that there existed more than 6 customers in a vehicle.

The algorithm can be formally stated as follows (VNSIRP):

Step1 *Define k_{max}, the maximum number of neighbourhood structure to be explored.*

Step2 *Construct an initial solution by finding a set of initial routes with pick-up quantities that exactly match the demand in every period. Let this solution be represented by x. The initial routes are formed as follows:*

2.1 *For each period j, $j = 1,2,\ldots,T$, arrange the suppliers $\{s_1, s_2, \ldots, s_N\}$ and the assembly plant $\{s_{N+1}\}$ around the depot (s_0) such that the y*

coordinate of the assembly plant is the same as the coordinate of the depot. Sort the suppliers $\{s_1, s_2, \ldots, s_N\}$ *in ascending order according to their new y coordinate values. Let* $s_{(i)}$ *be the i-th supplier after the sort. Set* $i = 1$ *and* $k = 1$. *Open a route* $R_k = \{\}$ *and set* $Q_k = 0$, *where* Q_k *is the total pick-up quantity assigned to cluster k.*

2.2 *If* $Q_k + d_{ij} \le C$, *assign* $s_{(i)}$ *to route* R_k, *set* $Q_k = Q_k + d_{ij}$ *and* $a_{ik} = d_{ij}$. *Otherwise, set* $k = k + 1$ *and open a new route* $R_k = \{\}$, *assign* $s_{(i)}$ *to* R_k. *Set* $a_{ik} = d_{ij}$ *and* $Q_k = a_{ik}$. *If* $i > m$, *set* $k = 1$ *and go to Step 2.3. Otherwise, set* $i = i + 1$ *and repeat Step 2.2.*

2.3 *{Routing} Sort the suppliers within route* R_k *according to their x-coordinate values in ascending order. Let* $s^k_{(i)}$ *be the* i^{th} *supplier in route* R_k *after the sort. If the x-coordinate of supplier N+1 is greater than or equals to the x-coordinate of supplier 0, form a route that starts at the depot* s_0, *visits suppliers* $s^k_{(1)}, s^k_{(2)}, \ldots, s^k_{(|R_k|)}, s_{N+1}$, *and returns to* s_0. *Otherwise, form a route that starts at supplier* s_0, *visits suppliers* $s^k_{(|R_k|)}, s^k_{(|R_k|-1)}, \ldots, s^k_{(1)}, s_{N+1}$ *and returns to* s_0. *Note that* $|R_k|$ *is the number of suppliers on route* R_k.

Step3 *Repeat the following until the stopping condition is met.*

3.1 *Set k=1*

3.2 *Until k=*k_{max}

3.2.1*Shaking: Generate a random solution, x' in the* k^{th} *neighborhood of* $x(x' \in N_k(x))$. *This is done in the following manner:*

i. Select randomly k number of suppliers from x. Note that the suppliers can be chosen from the same or different

periods.

ii. Identify the period where the supplier is to be moved. If the current period is t, then the selected period (to be moved) must be between 1and t−1. If suppliers I in period t, $t \in \{2,3,\dots\dots,T\}$ is chosen, then the amount a_{ilt} where $s_i \in R_l$ is aggregated with a_{iku} where $s_i \in R_k, u \in \{1,2,\dots,t-1\}$. If the total amount is greater than the vehicle capacity, $(a_{ilt} + a_{iku} > C)$, then select a new supplier. Otherwise, remove s_i from R_l in period t and do the following:

a. If $a_{ilt} + a_{iku} < C$ and $\sum_{p=1}^{|R_k|} a_{pku} + a_{ilt} \leq C$, update the pick-up amount for s_i in period u.

b. If $a_{ilt} + a_{iku} < C$ and $\sum_{p=1}^{|R_k|} a_{pku} + a_{ilt} > C$, perform GENI to insert s_i in other routes in period u. If this fails, create a new route for s_i and remove s_i from route R_k.

c. If $a_{ilt} + a_{iku} = C$ then remove supplier s_i from route $R_k$in period $u$and create its own route, s_0, s_i, s_{N+1}, s_0.

3.2.2 *Local Search*: *Perform GENI as local search.*

3.2.3 *Move or Not: If the local optimum is better than the incumbent, let $x = x$"and $k \leftarrow 1$: Otherwise set $k = k + 1$.*

The number of periods to be transferred was limited to at most two to ensure that the tradeoff between the savings in travelling distance and the increase in inventory cost was not too high. Note that the reassignment of the pick-up quantity could only be

done if the same supplier was visited in the period to be transferred. It was observed that if the periods to be transferred differed by more than two from the original period then the inventory costs were relatively higher and often produced inferior solutions. GENI was used to find local optima within a particular period. This ensured that the increase in inventory was offset by the savings in the travelling distance.

4.5.2 Illustration of Neighbourhood Structure

In this section, illustrations are given of the neighbourhood generated by the algorithm using a network consisting of 5 suppliers and 5 periods. Let the problem be represented as a binary matrix given in Figure 4.6a. The demand and collection matrices are given in Figures 4.6 (b and c) whilst the *x* and *y*-coordinate and the resultant distance matrix are given in Figures 4.6 (d and e).

		Period				
		1	2	3	4	5
Supplier	1	1	1	0	1	1
	2	1	1	0	1	1
	3	1	0	1	1	1
	4	1	1	1	0	0
	5	1	1	1	0	0

Figure 4.6a: Binary matrix representation.

		Period				
		1	2	3	4	5
Supplier	1	2	4	3	1	2
	2	2	3	3	1	2
	3	2	2	2	2	2
	4	3	1	2	4	3
	5	2	2	2	2	2

Figure 4.6b: Demand matrix.

		Period				
		1	2	3	4	5
Supplier	1	2	7	0	1	2
	2	2	6	0	1	2
	3	4	0	2	2	2
	4	3	1	9	0	0
	5	2	2	6	0	0

Figure 4.6c: Collection Matrix.

Assuming that the number of vehicles required and the route for each vehicle are given in Figure 4.6f, where the vehicle's capacity is fixed at 10 units.

As mentioned earlier, there are three types of neighbourhood that need to be considered. For easier understanding the neighbourhood is illustrated with an example.

		Coordinate	
		1	2
Depot		0	0
Supplier	1	1	4
	2	3	5
	3	6	5
	4	7	3
	5	4	0
A.Plant		5	2

Figure 4.6d: Coordinate of Depot, Assembly Plant, and Suppliers.

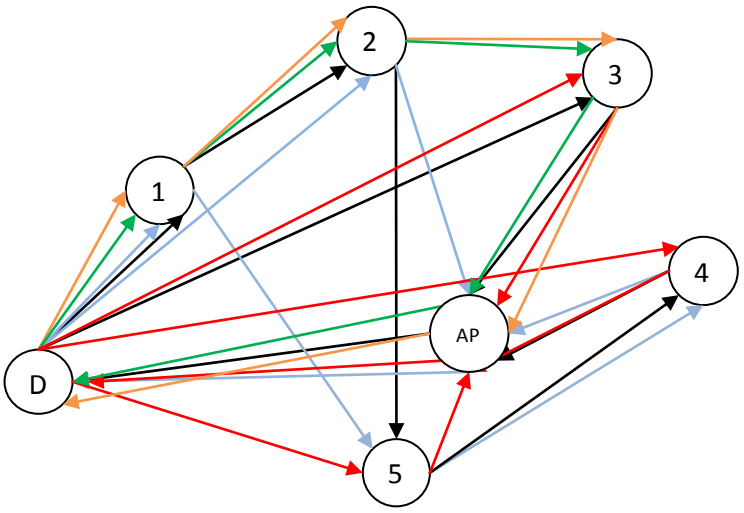
		Depot	Period					A.Plant
			1	2	3	4	5	
Depot		0						
Supplier	1	4.12	0					
	2	5.83	2.24	0				
	3	7.81	5.10	3.00	0			
	4	7.62	6.08	4.47	2.24	0		
	5	4.00	5.00	5.10	5.39	4.24	0	
A.Plant		5.39	4.47	3.61	3.16	2.24	2.24	0

Figure 4.6e: Distance matrix.

		Num. of Vehicles	Routes {Suppliers}	Pick-up amount (s)
Period	1	2	{1,2,5,4} ,{3}	{9},{4}
	2	2	{1,5,4} ,{2}	{10},{6}
	3	3	{2} ,{5} ,{4}	{6},{9},{2}
	4	1	{1,2,3}	{4}
	5	1	{1,2,3}	{6}

Figure 4.6f: The number of vehicles, routes and the total pick-up amounts for each vehicle.

Figure 4.7 (a), (b) and (c) illustrates the resultant networks that are formed based on the conditions considered in generating the respective neighborhood structure. Figure 4.7 depict the routes shown in Figure 4.6f.



i	**Supplier *i***
AP	Assembly Plant
D	Depot
Arrow : Routes for Period *j*	
→	Period 1
→	Period 2
→	Period 3
→	Period 4
→	Period 5

Figure 4.7: Network distribution for the matrix in Figure 4.6.

CASE 1:$a_{ilt} + a_{iku} < C$ and $\sum_{p=1}^{|R_k|} a_{pku} + a_{ilt} \leq C$

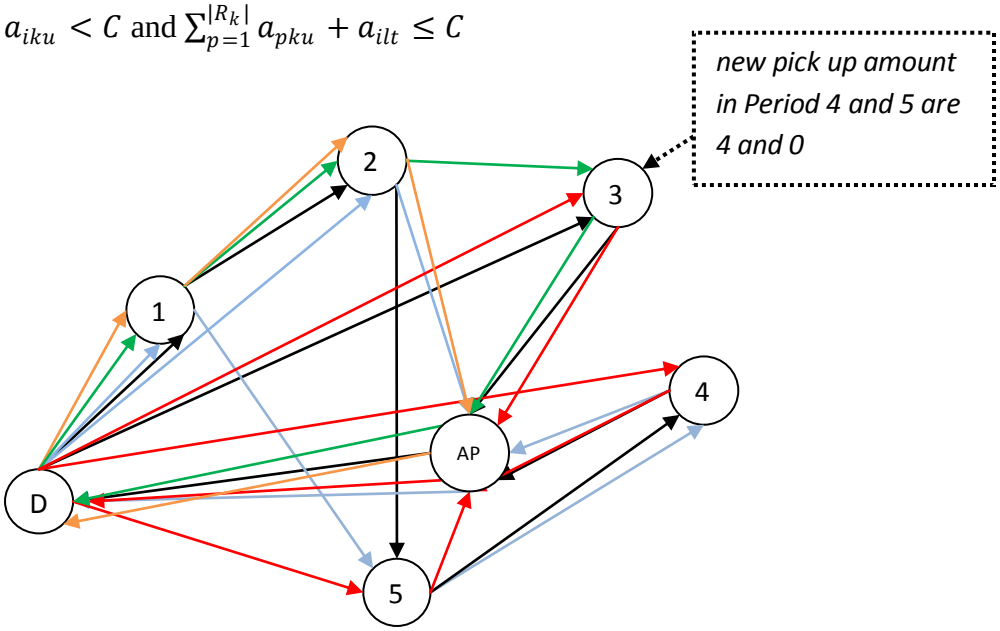


Figure 4.7a: An example of Case 1 where the supplier exists and the affected vehicle is not fully occupied.

In Case 1, both the aggregated amount and the total collection do not exceed the vehicle's capacity. Assume that Supplier 3 from Period 5 is selected and the amount to be transferred is 2 units. Since Supplier 3 is visited in Period 4, the total aggregated amount is 6 units and this does not exceed the vehicle's capacity. The new information

is updated and the total capacity in Period 5 is reduced to 4 units. This results in a reduced travelled distance at the expense of an increased in the holding cost.

CASE 2: $a_{ilt} + a_{iku} < C$ and $\sum_{p=1}^{|R_k|} a_{pku} + a_{ilt} > C$

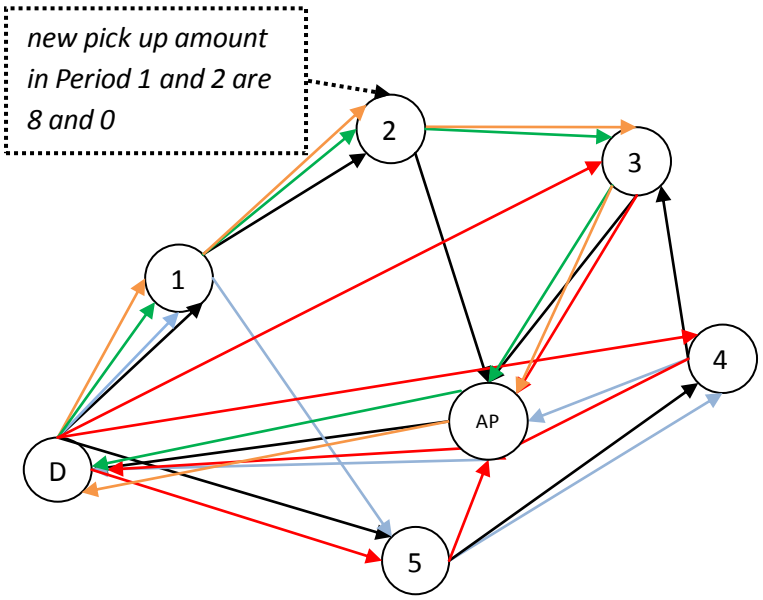


Figure 4.7b: An example of Case 2 where the transfer is merged but the total amount of the vehicle exceeds its capacity.

In Case 2, the aggregated amount is less than a vehicle's capacity but the total amount exceeds the capacity constraint. Assume that Supplier 2 from Period 2 which has a pick up amount of 6 units is selected to be removed. The aggregated amount in Period 1 is 8 units; however, the total amount on vehicle 1 is 15 units which exceed the vehicle's capacity. Supplier 2 will be removed and inserted in the other vehicles within the same period. Since vehicle 2 cannot accommodate anymore units (it had reached a maximum capacity), then a new vehicle is initiated with a new route $0 - 2 - AP - 0$.

CASE 3: $a_{ilt} + a_{iku} = C$

In Case 3, the amount in Period 4 is transferred to Period 3 for Supplier 1. Insertion in the existing vehicles in Period 3 is done by using GENI (or Least Insertion

Method). Note that in this example, the least possible insertion method is used as the GENI method needs at least 3 nodes to apply. Let's say the least insertion is in vehicle 1 with new route [D → 1→5→AP→D] and new pick-up amounts of 7 units.

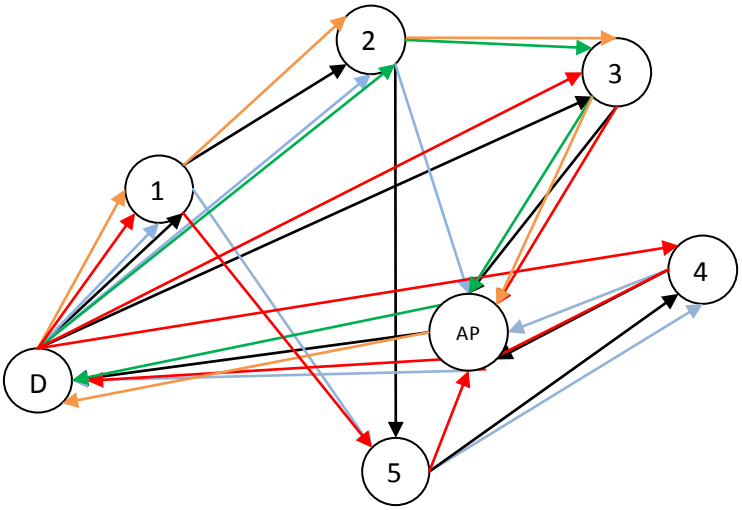


Figure 4.7c: An example of Case 3 where the amount collected matches the vehicle's capacity

4.5.3 Genetic Algorithm (GA)

The same types of representations that have been used as a comparison to the algorithm, VNS are presented. This work has been carried out by Aziz and Moin (2007).

Binary Matrix Representation

The representation represents a chromosome as a binary matrix of size where N is the number of suppliers while T defines the number of periods. Let

$$Ch^{l}(i,j)=\begin{cases}1 & \text{if Chromosome } l \text{ has Supplier } i \text{ visited in period } j\\ 0 & \text{otherwise}\end{cases}$$

It is noted that the amount to be collected depends on whether there will be a collection in the subsequent period or not. Since backordering is not allowed, the total

collection from supplier i in period j is the sum of all the demands in period $j, j+1,\ldots,k-1$ where the next collection will be made in period k. As the initial inventory, I_{i0} for $i = 1,2,\ldots,N$ is assumed to be zero, the values in the first column consist of all ones. However, the algorithm can be adjusted accordingly if the initial inventory for part i is given or known in advance. If all the initial inventories exceed the first period demands for all the suppliers, then the first column in the chromosome can be generated randomly.

Figure 4.8 illustrates an example of the chromosome representation for a problem with 5 suppliers and 5 periods with the demand matrix given in Figure 4.8(a). For instance in period 2, only suppliers 4 and 5 are visited as shown in Figure 4.8 (b). Note that in column 1, all suppliers are visited as expected (due to the assumption that no backlogging is allowed and the initial starting inventory is zero for all suppliers), i.e. $ch_{i1} = 1, \forall i = 1,\ldots,5$. It can be seen that supplier 1 is visited in period 1 and period 3 where a collection of 4 and 8 units is performed respectively (see Figure 4.8 (c)) with a resulting inventory as shown in Figure 4.8 (d).

Crossover Operator

The authors employ a two dimensional uniform crossover (modified to suit the matrix representation) where a binary mask of size $(N \times T)$ is generated randomly for each pair of parents. The position of the ones in the binary mask determines the values in the first parent that are transferred to the first offspring and the elements in position zeros are obtained from the second parent. A complimentary mask is used to deduce the second offspring. The modified uniform crossover is presented in Figure 4.9. Note that in both children, period 1 is not affected as all suppliers need to be visited.

For instance, as a value of zero is assigned to crossover mask for supplier 1 in period 2, this Child 1 will inherit the gene from Parent 2, which is 1. It is noted that the numbers in italic are inherited from Parent 2.

Supplier	Period 1	2	3	4	5
1	2	2	4	2	2
2	2	2	4	2	2
3	2	3	1	3	2
4	1	1	1	1	1
5	4	1	2	1	4

Figure 4.8a: Demand matrix.

Supplier	Period 1	2	3	4	5
1	1	0	1	0	0
2	1	0	0	1	0
3	1	0	1	0	1
4	1	1	1	0	1
5	1	1	0	0	1

Figure 4.8b: Binary matrix representation.

Supplier	Period 1	2	3	4	5
1	4	0	8	0	0
2	8	0	0	4	0
3	5	0	4	0	2
4	1	1	2	0	1
5	4	4	0	0	4

Figure 4.8c: Collection matrix.

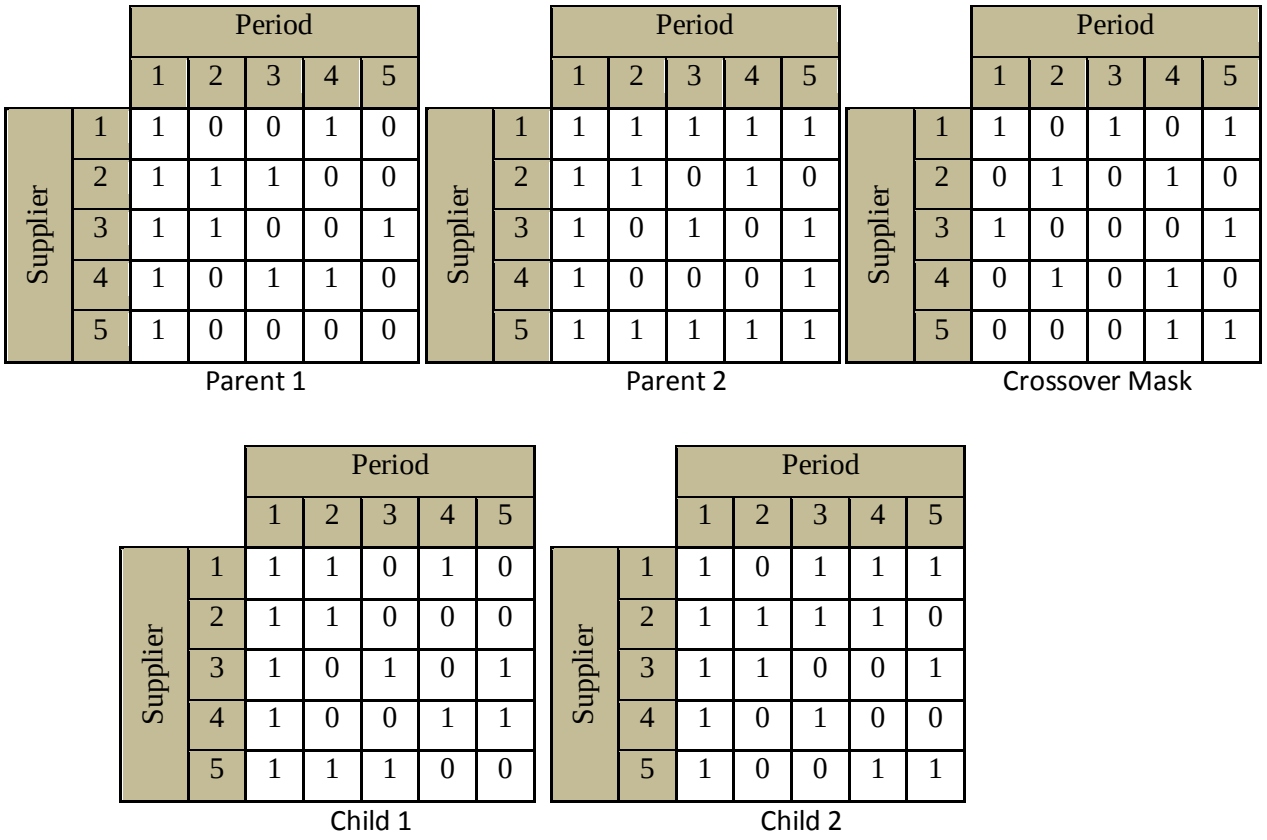
Supplier	Period 1	2	3	4	5
1	2	0	4	2	0
2	6	4	0	2	0
3	3	0	3	0	2
4	0	0	1	0	0
5	0	3	1	0	0

Figure 4.8d: Inventory matrix.

Mutation Operator

Mutation is a genetic operator, analogous to the biological mutation, which is used to maintain genetic diversity from one generation of a population of chromosomes to the next. The classic example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. The purpose of mutation in GAs is to allow the algorithm to avoid local minima by preventing the

population of chromosomes from becoming too similar to each other, thus slowing or even stopping the evolution. This reasoning also explains the fact that most GA systems tend to avoid taking the fittest of the population only when generating the next chromosome but use rather a random selection (or pseudo-random with a weighting towards those that are fitter).



Parent 1	1	2	3	4	5
1	1	0	0	1	0
2	1	1	1	0	0
3	1	1	0	0	1
4	1	0	1	1	0
5	1	0	0	0	0

Parent 2	1	2	3	4	5
1	1	1	1	1	1
2	1	1	0	1	0
3	1	0	1	0	1
4	1	0	0	0	1
5	1	1	1	1	1

Crossover Mask	1	2	3	4	5
1	1	0	1	0	1
2	0	1	0	1	0
3	1	0	0	0	1
4	0	1	0	1	0
5	0	0	0	1	1

Child 1	1	2	3	4	5
1	1	1	0	1	0
2	1	1	0	0	0
3	1	0	1	0	1
4	1	0	0	1	1
5	1	1	1	0	0

Child 2	1	2	3	4	5
1	1	0	1	1	1
2	1	1	1	1	0
3	1	1	0	0	1
4	1	0	1	0	0
5	1	0	0	1	1

Figure 4.9: Crossover Operator.

The classical mutation operator is applied in this research where a random variable is generated for each bit in a sequence. If this random number is found to be less than the specified mutation rate, then the value of that particular bit will be mutated (in this case the value of one will be changed to a zero and *vice versa*).

The genetic algorithm for the IRP can generally be divided into four steps which include: (i) the generation of the initial population, (ii) the evaluation of the objective function that comprises the decoding of the chromosome and applying the double sweep algorithm (Lee *et al.*, 2003) to cluster and route the suppliers, (iii) the application of the

genetic operators, and finally (iv) the formation of a new population. This process iterates until a suitable stopping criterion is met.

4.6 Computational Results

4.6.1 Data

The data sets taken from Lee *et al.* (2003) are adapted and modified. The original four data sets were S12T14, S20T21, S50T21 and S98T14 that comprised of (12 suppliers, 14 periods), (20 suppliers, 21 periods), (50 suppliers, 21 periods) and (98 suppliers, 14 periods), respectively. From these data sets, other ten data sets were created by varying the number of periods in each planning horizon to represent small, medium and large size problems. The 10 datasets are S12T5, S12T10, S20T5, S20T10, S20T14, S50T5, S50T10, S50T14, S98T5 and S98T10. The locations of the suppliers for S12T14, S20T21 and S50T21 were generated randomly in a square of 100×100. The locations of the suppliers for the S20T14 were extended from the S12T14 data sets by adding 10 new suppliers. Similarly, the S50T21 suppliers were extended from the S20T21 locations and randomly generating the locations of an additional 30 suppliers. The S98T14 was based on a real life data and the suppliers were closely located.

All the data sets, with the exception of S50T21, had demands in every period. Some suppliers in the S50T21 data set may not received the demand for their products until the later periods. As some of the data sets were extracted from S50T21, it was possible that the demand for some of the products was zero. Data sets S50T5 and S50T10 consisted of products with zero demand. The suppliers of these products could be effectively eliminated from the representation as they would never be visited in the planning horizon.

It is noted that the demands for S98T14 were given in real values and the amount varied significantly between each product. The cost per unit distance, fixed cost

and the vehicles' capacity were increased to 50, 200 and 400 respectively. It is also noted that the demand for each product from one period to the other for this data set was constant as this was a common feature in the automotive industry.

Table 4.1 tabulates the characteristics of the data sets. Note that the depot is located at $(0,0)$ for all the data sets and there is no restriction on the number of stops for each trip. The scatter plots of the locations of suppliers for each data set are shown in Figure 4.10a, Figure 4.10b, Figure 4.10c, and Figure 4.10d respectively. We noted that for all cases S12, S20, S50 and S98; the data given is used for all periods: T5 (5 periods), T10, T14 and T21 (if exist). We also noted that the coordinate of the assembly plant for caseS98 was different compared to the Lee *et al.*(2003) as the coordinate given fails to route and violated the route length. The data set is given in Appendix B.

Table 4.1: Characteristics of the data sets.

Case	**S12**	**S20**	**S50**	**S98**
Fixed Cost (F)	20	20	20	200
Cost per Unit Travelling Distance	1	1	1	50
Data Set(Lee *et al.* (2003))	S12T14	S20T21	S50T21	S98T14
Vehicle Capacity	10	10	10	400
Maximum Route Length	140	140	140	150
Range of Holding Costs	[3,27]	[3,27]	[1,9]	[1,44]
Range of Demand for Each Product*	[1,4]	[1,4]	[0,9]	[0.0400, 393.3300]
Coordinate of Assembly Plant	(10,20)	(10,20)	(10,20)	(42.31, -83.17)

* See Appendix B.

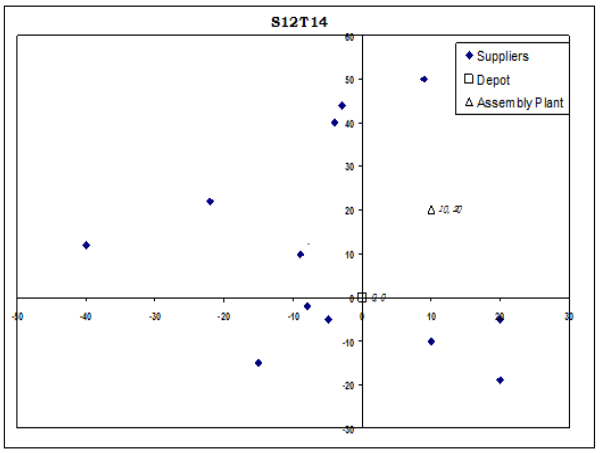


Figure 4.10a: Scatter plot for S12T14.

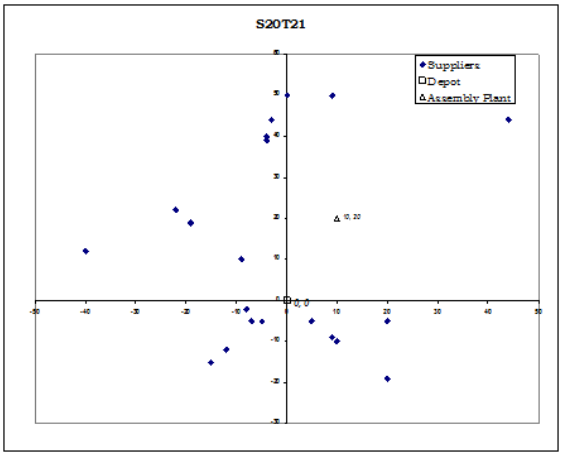


Figure 4.10b: Scatter plot for S20T21.

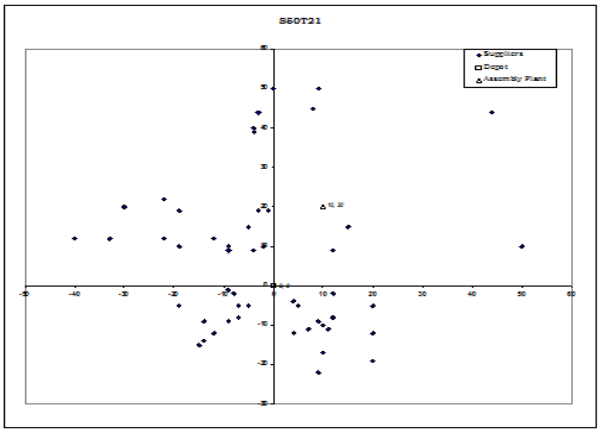


Figure 4.10c: Scatter plot for S50T21.

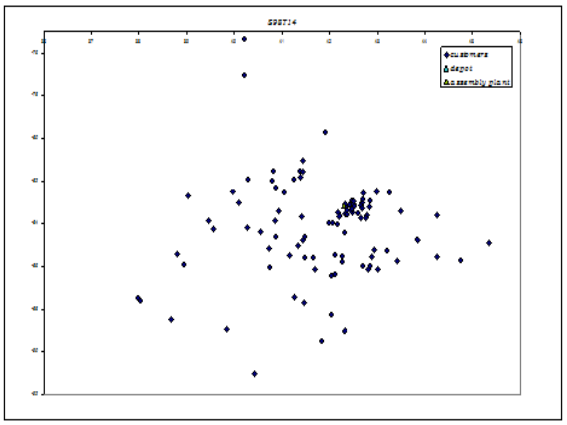


Figure 4.10d: Scatter plot for S98T14.

4.6.2 Results and Discussion

The algorithms for VNS were written in MATLAB whilst the GA was written in Microsoft Visual C++ using Genetic Algorithms Library (GAlib) to run the program.

For GA, the number of generations, crossover rate and mutation rate were fixed at 300, 0.9 and 0.01 respectively for all the problems. The population size was fixed at 200 individuals (extended to 500 for S98). Each data set was executed ten times for each algorithm.

For VNS, VNSIRP algorithm embedded GENI as a local search. The maximum number of neighbourhood or k_{max} was set at 30. Based on the limited experiments, it was observed that if it were too low (eg: 10 or 20) one tended to get trapped in the local

search, and if it were too high (e.g.: 50 until 120), the final results generated were comparable to $k_{max} = 30$. The number of changes that can be applied cannot exceed $max\_Period - 1 \times max\_Supplier$, where max_Period was the maximum number of periods, and $max\_Supplier$ was the maximum number of suppliers respectively. Note that all suppliers will receive delivery in the first period. The local search, GENI method was applied by considering insertion in both orientations. The number of neighbours in each N_k was limited to 10.

Table 4.2 presents the best results based on the total cost for each data set. The mean, standard deviation and the percentage difference between the best solutions for each data set is presented in Table 4.3. In Table 4.2 it is observed that VNS produced better results for all cases except for case S98T10. For small cases S12, VNS were able to give results in good *cpu* time. However, for bigger cases we observed that longer time is needed to obtain the results.

Table 4.2: Characteristics of the best results for 14 data set.

DATA SET	Total Cost	Total Distance	Inventory Cost	# Vehicles	CPU time
S12T5					
GA	2813.6	1748.6	765	15	1033
VNS	**2116.69772**	1575.69772	261	14	184.59598
S12T10					
GA	5138.7	3852.7	666	31	1422
VNS	**4400.44429**	3411.444293	369	31	437.66081
S12T14					
GA	6463.9	4843.9	780	42	1253
VNS	**6301.09489**	4903.094894	498	45	533.86662
S20T5					
GA	4019.9	2543.9	996	24	977
VNS	**3214.66361**	2389.663606	345	24	1200.7709
S20T10					
GA	9530.5	6181.5	2289	53	1960
VNS	**6689.00211**	5019.002108	690	49	2384.88089

Table 4.2: Continued.

DATA SET	Total Cost	Total Distance	Inventory Cost	# Vehicles	CPU time
S20T14					
GA	10921	7733.4	1788	70	3050
VNS	**9575.98166**	7210.981663	1005	68	6901.06304
S20T21					
GA	15254	11239	1935	104	3651
VNS	**14498.3236**	10953.32362	1425	106	6029.84425
S50T5					
GA	5729.6	4336.6	453	47	2559
VNS	**5448.79888**	4258.798879	270	46	15920.9785
S50T10					
GA	12081.1	8868.1	1213	100	3630
VNS	**11493.5814**	8908.581376	545	102	57054.6513
S50T14					
GA	17521	12727	1934	143	7405
VNS	**16699.2071**	13089.20708	650	148	70968.0685
S50T21					
GA	27189	19508	3281	220	8708
VNS	**25520.636**	20015.63603	1005	225	47699.8954
S98T5					
GA	622282.2	12055.62789	6900.85	63	6211.688
VNS	**614969.003**	12001.76965	2.08E+03	64	4795.377
S98T10					
GA	**1227730**	23618.67115	22196.24	123	20830.47
VNS	1238072.3	24184.95692	3024.50000	129	11125.742
S98T14					
GA	1748293	33642.90853	31147.38	175	40951.13
VNS	**1737358.2**	33953.80804	3467.7500	181	24589.923

Table 4.3: Results in terms of best total cost, mean, standard deviation and percentage for the 14 data sets.

DATA SET	Best	Mean	Standard Deviation	Percentage Difference
S12T5				
GA	2813.6	2874.05	33.75240732	32.92403
VNS	**2116.69772**	2220.682552	53.46672405	
S12T10				
GA	5138.7	5254.36	69.75471629	16.77684
VNS	**4400.44429**	4602.540375	99.8780332	
S12T14				
GA	6463.9	6524.57	62.0473305	2.583759
VNS	**6301.09489**	6518.409439	118.7918287	
S20T5				
GA	4019.9	4130.33	64.64047666	25.04885
VNS	**3214.66361**	3291.6613	47.48976733	
S20T10				
GA	9530.5	9760.74	139.1531067	42.48015
VNS	**6689.00211**	6905.192298	110.2097406	
S20T14				
GA	10921	11339.43	267.8343727	14.04575
VNS	**9575.98166**	9889.577845	175.1161491	
S20T21				
GA	15254	15572.9	167.2951152	5.212164
VNS	**14498.3236**	14794.60764	174.1445678	
S50T5				
GA	5729.6	5808.87	57.28982943	5.15345
VNS	**5448.79888**	5556.953049	60.05909385	
S50T10				
GA	12081.1	12342.23	156.4574848	5.11171
VNS	**11493.5814**	11779.95515	196.7903542	
S50T14				
GA	17521	17727.9	127.3529217	4.921149
VNS	**16699.2071**	16904.2173	126.647619	
S50T21				
GA	27189	27415.6	160.1653312	6.537314
VNS	**25520.636**	25957.79358	267.1046518	

Table 4.3: Continued.

DATA SET	Best	Mean	Standard Deviation	Percentage Difference
S98T5				
GA	622282.2	627889.2	6216.909	1.189198
VNS	**614969.0026**	621393.77	5625.8505	
S98T10				
GA	**1227730**	1251571	12671.58	0.842396
VNS	1238072.346	1243025.22	4762.692796	
S98T14				
GA	1748293	1760186	11384.16	0.629395
VNS	**1737358.152**	1744193.9	3774.7856	

4.7 Enhanced Variable Neighbourhood Search (EVNS)

Some modifications are introduced to improve the results obtained. A pre-optimization step (Step 2a) is introduced after the clustering and routing was performed by the double sweep algorithm (DSA). In addition a different neighbourhood structure was introduced in search for a better solution as selecting the best neighbourhood structure was very crucial. In the first algorithm, GENI was performed only during an insertion procedure if the sum of the pick-up quantity for the respective supplier exceeded the vehicle's capacity. Performing the DSA produced deterministic results where there were not many variations between solutions, resulting in not much improvement in the overall solutions obtained.

In order to improve the solution, in Step 3.2.1 part ii (b) contraction was performed to identify the suppliers to be removed from the route prior to performing GENI. The function of a contraction was to contract the feasible solution by removing one or more suppliers on the route in order to accommodate the additional increase in the total pickup quantity for supplier s_i. The supplier to be removed was selected among the suppliers contributing the most to the transportation cost from a poorly utilized route

(Lee *et al.*, 2003). The quality of a route was measured by the ratio of the length of the route to the total pickup quantity. A long route with a small pickup quantity was considered poorly utilized and was very likely to be removed. The ratio was calculated for all the suppliers on route R_k and the supplier with the maximum ratio was selected to be removed. The selected supplier to be removed would be inserted in other routes in period u using GENI.

The second modification was made by adding the selected supplier, if it was not visited in the preceding two periods. Again GENI (or the Least Cost Method if the number of suppliers was less than 3 on the existing route) was utilized to insert the supplier in the existing routes. If this failed due to the violation of constraints, a new route would be initiated. Finally, the local search was modified where a simple nearest neighbour method was employed to replace the more time consuming GENI.

Step 3.2. is modified as follows:

Step 3.2:*Until k=*k_{max}

3.2.1 *Shaking: Generate a random solution, x' in the* k^{th} *neighborhood of* $x(x' \in N_k(x))$*. This is done in the following manner:*

i. Select randomly k number of suppliers from x. Note that the suppliers can be chosen from the same or different periods.

ii. Identify the period where the supplier is to be moved. If the current period is t, then the selected period (to be moved) must be between 1and t−1. If suppliers I *in period* $t, t \in \{2,3,\dots\dots,T\}$*is chosen, then the amount* a_{ilt} *where* $s_i \in R_l$ *is aggregated with* a_{iku} *where* $s_i \in R_k, u \in \{1,2,\dots,t-1\}$*. If the total amount is greater than a vehicle's capacity,* $(a_{ilt} + a_{iku} > C)$*, then select a new supplier. Otherwise,*

remove s_i from R_l in period t and do the following:

(a) If $a_{ilt} + a_{iku} < C$ and $\sum_{p=1}^{|R_k|} a_{pku} + a_{ilt} \leq C$, update the pickup amount for s_i in period u.

(b) If $a_{ilt} + a_{iku} < C$ and $\sum_{p=1}^{|R_k|} a_{pku} + a_{ilt} > C$,

- *Update the pick-up amount for s_i in period u.*
- *Perform contraction by choosing the maximum ratio of $r_i = c_{pi} + c_{im} - c_{pm}/a_{it}$, where p immediately precedes i, supplier m immediately succeeds supplier i and a_{it} the pickup quantity of supplier i.*
- *Remove the selected supplier and insert in other routes using GENI. If no route can accommodate the supplier due to violation of constraints, initiate a new vehicle.*

(c) If $a_{ilt} + a_{iku} = C$ then remove supplier s_i from route R_k in period $u$and create its own route, s_0, s_i, s_{N+1}, s_0.

(d) If s_i is not visited in the two preceding periods, add s_i to the tour and insert into one of the existing vehicles using Least Cost Method. If this fails due to violation of constraints, initiate a new vehicle.

3.2.2 *Local Search*: *Perform nearest neighbor as local search.*

3.2.3 *Move or Not*: *If the local optimum is better than the incumbent, let $x = x$" and $k \leftarrow 1$: Otherwise set $k = k + 1$*

4.7.1 Computational Results

In order to compare with the analytical results using the mathematical programming software CPLEX version 11.1, the route length constraint (constraint (1j))

was removed from the formulation given in Section 4.3 and reformulated using the network flow approach where the index for the number of vehicles is omitted. CPLEX was not able to produce significant solutions if the route length was not removed. The new formulation is given as follows:

$$Z = \min \underbrace{\sum_{i \in S} h_i \left(\sum_{t \in \tau} I_{it} \right)}_{(A)} + \underbrace{V \left(\sum_{\substack{j \in S \\ j \neq i}} \sum_{i \in S \cup D} c_{ij} \left(\sum_{t \in \tau} x_{ijt} \right) + \sum_{i \in S} c_{i,N+1} \left(\sum_{t \in \tau} x_{i,N+1,t} \right) \right)}_{(B)} + \underbrace{\left(F + c_{N+1,0} \right) \sum_{t \in \tau} \sum_{i \in S} x_{0it}}_{(C)} \quad (2a)$$

Subject to:

$$I_{it} = I_{i,t-1} + a_{it} - d_{it}, \quad \forall i \in S, \forall t \in \tau \quad (2b)$$

$$\sum_{\substack{i \in S \cup D \\ i \neq j}} q_{ijt} + a_{jt} = \sum_{\substack{i \in S \cup P \\ i \neq j}} q_{jit}, \quad \forall j \in S, \forall t \in \tau \quad (2c)$$

$$\sum_{i \in S} q_{i,N+1,t} = \sum_{i \in S} a_{it}, \quad \forall t \in \tau \quad (2d)$$

$$\sum_{\substack{i \in S \cup D \\ i \neq j}} x_{ijt} = \sum_{\substack{i \in S \cup P \\ i \neq j}} x_{jit}, \quad \forall j \in S, \forall t \in \tau \quad (2e)$$

$$\sum_{j \in S} x_{ijt} = \sum_{j \in S} x_{jkt}, \quad i \in D, k \in P, \forall t \in \tau \quad (2f)$$

$$q_{ijt} \leq C \cdot x_{ijt}, \quad \forall i \in S, \forall j \in S \cup P, i \neq j, \forall t \in \tau \quad (2g)$$

$$I_{it} \geq 0, \quad \forall i \in S, \forall t \in \tau \quad (2h)$$

$$a_{it} \geq 0, \quad \forall i \in S, \forall t \in \tau \quad (2i)$$

$$x_{ijt} \in \{0,1\}, \quad \forall i, j \in S, \forall t \in \tau \quad (2j)$$

$$x_{ijt} = 0, \quad i \in S, j \cup D, \forall t \in \tau \quad (2k)$$

$$x_{ijt} = 0, \quad i \in P, j \cup S, \forall t \in \tau \quad (2l)$$

$$q_{ijt} \geq 0 \ \forall i \in S, \forall j \in S \cup P, \forall t \in \tau \quad (2m)$$

$$q_{0it} = 0 \ \forall i \in S, \forall t \in \tau \quad (2n)$$

It is noted that all the previous notations remain the same except for x_{ijt} and q_{ijt} which represent the number of times that the directed arc (i, j) is traversed by vehicles in period t and the quantity transported through the directed arc (i, j) in period t respectively.

The objective function (2a) comprises both inventory costs and the transportation cost (variable travel costs and vehicle fixed cost. The fixed transportation cost consists of fixed cost incurred per trip and the constant cost of vehicles returning to the depot from assembly plant. Constraint (2b) is the inventory balance equation for each product at assembly plant. (2c) is the product flow conservation equations, assuring the flow balance at each supplier and eliminating all subtours. (2d) assures the accumulative picked-up quantities at the assembly plant and (2e) and (2f) ensure that the number of vehicles leaving a supplier, assembly plant or the depot is equal to the number of its arrival vehicles. Note that constraint (2f) is introduced because each vehicle has to visit the plant before returning to the depot. (2g) guarantees that the vehicle capacity is respected and gives the logical relationship between q_{ijt} and x_{ijt} which allows for split pick-ups. (2h) is to make sure that the demand at the assembly plant is completely fulfilled without backorder. The remaining constraints are the nonnegativity constraints.

4.7.2 Selection of Neighborhood

The number of maximum neighborhood structure explored was given by $k_{max} = \min\left\{\frac{1}{4}(T \times N), 100\right\}$ where T and N are the maximum number of periods and the maximum number of suppliers respectively. This is to ensure that the algorithm do not waste a lot of computational time unnecessarily. Number of neighbors used in each k^{th} neighborhood structure is 100, it is chosen based on the limited number of experiments carried out. It is noted that if the number of neighbors explored was too

small, the results obtained are poor as there is too little variation and it cannot escape from the local optimum. And if it was too big a lot of time was spent exploring neighbors that are further away from the incumbent solution. Additional condition was also imposed on the stopping criteria where the time spent between two successive improvements is set to 60 minutes. In other words if there were no other improvements the programme was terminated and the last solution found was accepted as the best solution.

4.7.3 Results and Discussions

It is noted that EVNS1 incorporated *2-opt* in the pre-optimization stage and in local search whilst EVNS2 incorporated*3-opt* in the pre-optimization stage and in the local search. The algorithms were run for 10 times each for all the 14 test cases with 100 members and CPLEX were run for a minimum of 3600 seconds. The results for VNSIRP1 and VNSIRP2 are shown in Table 4.4 where the best results for VNSIRP1 and VNSIRP2 were compared with the lower bound and the best integer found obtained using CPLEX software. CPLEX fails to find significant solutions for S50T21, S98T10 and S98T14. The best results using EVNS are given in bold. The results for EVNS2 were not better compared to EVNS1. And it was also observed that as the number of suppliers in each vehicle was small, not much improvement could be benefitted from *3-opt*.

Detailed results using the number of neighborhood equals to 100 and 50 are given in Table 4.5. The results are the best of 10 runs and they are significantly better using $N_k = 100$ compared to $N_k = 50$. It is shown that a too low number of neighbours will be trapped in Local Search. This is however at the expense of higher computational times. The best results are given in bold.

Table 4.4 : Results for Lower Bound, EVNS1 with 2 –opt and also EVNS2 with *3-opt*.

Data	Lower Bound	Best Integer			Gap ((BI-LB)/LB)(%)	EVNS1 (*2 opt*)	# veh	Time (secs)	EVNS2 (*3 opt*)	# veh	Time (secs)
		Objective	# of vehicles	Time (secs)							
S12T5	1819.1115	1885.2965	14	2340	3.51%	**2076.09**	14	78.5153	2101.048	15	66.47203
S12T10	3634.0350	3752.6392	27	3600	3.16%	**4280.83**	30	280.9578	4349.239	31	194.2369
S12T14	5350.2538	5556.5428	39	3600	3.71%	**6111.22**	44	605.50228	6216.60	45	610.8843
S20T5	2790.2284	2883.6062	21	3600	3.24%	**3201.7**	21	425.99193	3209.261	22	406.367
S20T10	5601.1254	5919.1889	44	3600	5.37%	**6591.03**	45	1494.92638	6627.951	46	1319.612
S20T14	7829.6439	8602.6161	64	2580	8.99%	**9289.89**	64	2778.53381	9386.829	65	3074.78
S20T21	11771.7433	12779.0943	94	2520*	7.88%	14228.2	97	4737.98437	**14215.68**	99	4916.559
S50T5	4735.3188	5425.1222	46	3600	12.71%	**5390.36**	47	2352.71348	5464.526	47	2226.883
S50T10	9815.8714	10863.7864	99	3600	9.65%	**11413.5**	101	5189.88807	11699.09	101	4136.476
S50T14	14017.6983	16618.3996	143	3600	5.65%	**16344.9**	142	7589.77625	16467.00	144	7649.415
S50T21	21485.253837	No solution given		3600		**25248.5**	223	17230.24805	25415.48	221	15392.54
S98T5	548104.9499	581951.8482	58	7200	5.82%	**605655.2**	63	5769.12	623726.7	65	6630.078
S98T10	1095020.4528	9582506.321*			88.57%	**1211830**	126	18739.01	1230213.92	128	9611.00321
S98T14	**					**1718750**	179	48079.18	1728678.64	180	51529.71

* Incomplete optimization.

** The programmes fail to produce any feasible solution.

Table 4.5 : The results run for all 14 cases using EVNS1, with different number of neighbours.

num_of_neigh=	100

	Total cost	Vehicle	Holding Cost	Distance	Time
S12T5	**2076.0939**	14	174	1622.093916	78.5153
S12T10	**4280.8289**	30	312	3368.828883	280.9578
S12T14	**6111.2243**	44	381	4850.224312	605.5023
S20T5	**3201.7007**	21	270	2511.700729	425.9919
S20T10	**6591.0328**	45	450	5241.032816	1494.926
S20T14	**9289.8891**	64	849	7160.889131	2778.534
S20T21	14228.16	97	996	11292.15995	4737.984
S50T5	**5390.3574**	47	185	4265.357386	2352.713
S50T10	**11413.509**	101	439	8954.509383	5189.888
S50T14	**16344.866**	142	649	12855.86649	7589.776
S50T21	**25202.282**	223	798	19944.28236	15544.24
S98T5	**605655.16**	63	2.46E+03	11811.89355	5769.12
S98T10	**1211830.4**	126	4.48E+03	23643.09438	18739.01
S98T14	1718749.5	179	5.02E+03	33558.62773	48079.18

num_of_neigh=	50

	Total cost	Vehicle	Holding Cost	Distance	Time
S12T5	2132.490616	15	117	1715.490616	40.06106
S12T10	4417.578385	30	417	3400.578385	154.285
S12T14	6233.891105	45	363	4970.891105	280.4118
S20T5	3234.785508	22	243	2551.785508	187.092
S20T10	6690.759043	46	459	5311.759043	713.9854
S20T14	9448.999249	65	747	7401.999249	1514.645
S20T21	**14218.08372**	100	1089	11129.08372	3554.686
S50T5	5494.303073	48	149	4385.303073	1200.662
S50T10	11546.9774	102	417	9089.977396	4006.542
S50T14	16652.11541	145	368	13384.11541	4917.822
S50T21	25618.02256	225	683	20435.02256	8641.535
S98T5	614720.84	64	1.35E+03	12011.4168	4416.33
S98T10	1229395.68	128	3.29E+03	24010.1136	11234.006
S98T14	**1702925.976**	177	7.45+E03	33201.43952	26039.499

In this research, the entire VNS algorithm for all cases was coded using MATLAB 7.5 and run on Intel Core 2 Duo processor with 4Ghz speed while GA was coded using C++ software and GALib for the GA operators. To obtain the results, the data set was executed not less than ten times.

4.8 Conclusion

As a conclusion, selecting of neighbourhood structure was very crucial and had to be suitable with the problems. For a smaller problem, it was easy to choose. But for a bigger problem, a lot of considerations had to be made, for example less complicated structure, not too complex and hard to apply structure. However, a too simple structure would not give a good solution either.

Choosing the correct local search was also very crucial and had to be suitable with the problem. A very complicated local search used in this research, GENI was good with an increase in the running *cpu* time. Applying a more complicated local search *3-opt*did not provide the expected results as the number of vehicles used in a trip was not much to give a variety in the search.

Applying *2-opt* as the pre optimization and also in the local search was shown to be the best way in solving IRP. Along with it, was the use of GENI as the fundamental in the neighbourhood structure. However, the results obtained were quite far compared to the best objective found by CPLEX.